矩阵和广义表

2019年4月14日 21:30

-
- 矩阵

1. 主序major order
   1) C等语言都是行主序，&a[i][j]=&a[0][0]+(jmax*i+j)*单元素大小
   2) FORTAN是列主序，可以把ij互换，imax和jmax也互换
2. 稀疏矩阵：稀疏因子δ=t/m/n<=0.05的m行n列矩阵
   1) 三元组顺序表法
      i.
```
typedef  struct {
        int i, j;
        Elemtype e;
}Triple;
```
      ii.
```
typedef  struct {
        Triple data[MAXSIZE];
        int mu, nu, tu;
}TSMatrix;
```
      iii.
```
void sum( TSMatrix *A, TSMatrix *B, TSMatrix *C){//C=A+B
        if(A->mu!= B->mu || A->nu!= B->nu){
                printf("非同型矩阵不能相加！");
                return;}
        C->mu= A->mu;
        C->nu= A->mu;
        C->tu= 0;
        int pa=1, pb=1;
        int ta= A->tu, tb= B->tu;
        Triple *da= A->data, *db= B->data;
        while(pa<=ta && pb<=tb){
                if(++C->tu > MAXSIZE){
                        printf("MAXSIZE大小不足！");
                        return;}
                if(da[pa].i<db[pb].i ||
                da[pa].i==db[pb].i && da[pa].j<db[pb].j)
                        C->data[C->tu].e= da[pa++].e;
                else if(da[pa].i>db[pb].i ||
                da[pa].i==db[pb].i && da[pa].j>db[pb].j)
                        C->data[C->tu].e= db[pb++].e;
                else      //da[pa].i==db[pb].i && da[pa].j==db[pb].j
                        C->data[C->tu].e= da[pa++].e + db[pb++].e;}
        while(pa<=ta){
                if(++C->tu > MAXSIZE){
                        printf("MAXSIZE大小不足！");
                        return;}
                C->data[C->tu].e= da[pa++].e;}
        while(pb<=tb){
                if(++C->tu > MAXSIZE){
                        printf("MAXSIZE大小不足！");
                        return;}
                C->data[C->tu].e= db[pb++].e;}}
```

-
- 广义表

1. 广义表
    1) 广义表的元素可以是单个元素或另一个广义表，单个元素称为原子，另一个表称为其子表
    2) 第一个元素称为表头head，其他元素组成一个子表，该子表称为表尾tail
    3) ()意为空表，长度为0，表头表尾均为NULL
    4) (原子)长度为1，表头为该元素，表尾为NULL
    5) (())意为元素为空表的表，长度为1，表头为空表，表尾为NULL
2. 头尾存储法（头尾链表表示法）
    1) （头，尾1，尾2）的头是头，尾是（尾1，尾2），相当于把左括号移到第一个元素的后面
    2) tag表示表头是不是表结点
    3) tag=0时该结点是原子，没有表尾
    4) 所以表头是原子时，画图要先画一个tag为1的表，再让头指向tag为0的表
    5)
```
typedef struct GLNode{
    int tag; //0为原子结点，1为表结点
    union{
        elemtype data;
        struct sublist{
            struct GLNode *hp,*tp;
        }ptr;    // ptr.hp,ptr.tp指向表头、表尾
    };
}*Glist;
```
3. 孩子兄弟表示法（扩展线性链表表示法）
    1) tag为0时仍然可以有表尾，即，将头尾存储法中，原子结点的父结点的表尾搬到原子结点后，并置父结点的尾结点为空
    2) 指向原子表头的箭头也从头尾结点法的指向父结点改为指向tag为0的结点
    3)
```
typedef struct GLNode{
    int tag; // 0为原子结点，1为表结点
    union{
        elemtype data; //原子结点的值域
        struct GLNode *hp;      //表结点的表头指针
    }
    struct GLNode *tp;       // 指向下一个元素结点
}*Glist;
```